MANAGING REQUESTS FOR CONNECTION TO A SERVER

FIELD OF THE INVENTION

The present invention relates to a client/server and in particular a system and method for connecting clients to a heavily loaded server according to connection priority.

BACKGROUND

It is common knowledge that networks, such as the Internet, local area networks (LAN), and so forth have recently spread. In such networks, a server often executes various processes requested by client terminals through the network.

The allowable throughput of such a server has a limit, depending on the throughput of a CPU, the data transfer capacity at the input-output interface, the data transfer capacity between the database for storing various data and the server, the number of concurrent threads, the capacity of cache memory used in the processing of each thread, and so forth.

When a large number of requests are made in parallel from a plurality of client terminals through a network, the capacity of the server may be exceeded. As a result, it becomes difficult for the server to respond in a timely way, and in some cases the server will not be able to process requests from client terminals normally.

For this reason, the server limits the number of client terminals that can be connected. When the connection requests from client

terminals reach the limit, the server refuses to connect with any additional client terminals. Thus, the server ensures a certain level of throughput for executing processes requested by client terminals that have already been connected with the server, by refusing connection requests greater than the limit.

However, when the number of client terminals that can be connected is limited at the side of the server as described above, a user wishing to access the server may repeat a request for connection until a connection is established. For this reason, the server may receive far more connection requests than the number of client terminals actually requesting a connection. Furthermore, even when the server refuses a connection request, its processing load increases, as the refusal process is performed by use of a thread. Consequently, the server's responsiveness to connected clients may be reduced significantly by an avalanche of connection requests.

In addition, connections may be granted out of order, as connections are granted at random rather than according to waiting time.

SUMMARY

The present invention addresses the problems mentioned above. Accordingly, an object of the present invention is to provide a connection accepting system, an accepting server, and a connection-acceptance managing method, which are capable of reducing the processing load of the server and granting connections in the order of arrival of initial requests for connection.

To attain the aforementioned object of the present invention, a server in a connection accepting system of the present invention sets a connection priority for a client terminal and transmits

data concerning the connection priority to the client terminal, at the time of receiving a first connection request from the client terminal. Then, the server allows connection of the client terminal according to its connection priority at the time of receiving a second connection request from the client terminal after the first connection request. The client terminal displays connection priority information, based on the connection priority data transmitted from the server. In this manner, the server accepts a connection request from the client terminal with the highest connection priority.

The server may set the connection priority only in the case where a connection cannot be allowed at the time of receiving a first connection request from the client terminal. In the case where a connection can be allowed, the server can allow connection without setting any connection priority. In addition, at the time the first connection request is received from the client terminal, the connection priority can be set without condition regardless of whether or not a connection can be allowed. Based on this, connections can be managed.

The second connection request may be any request made after the first connection request. For example, if the connection request first made by the client terminal is assumed to be a first connection request, the third connection request and the requests thereafter, as well as the second connection request, can be thought of as the second connection request. Therefore, in the case where a connection request from the client terminal is refused when a connection cannot be established, a connection request made when a connection is allowed is considered to be the second connection request.

The connection priority that is set by the server may be indicated by a reference number, or a character or figure string or other symbols representing connection priority.

In addition, a connection request that is made to a server by the client terminal may be a connection request to that server. In the case where the connection accepting system of the present invention is further equipped with a second server for executing a process according to a request from the client terminal, the connection request may be a connection request to the second server. In this case, if the connection-management means of the first server allows connection of the client terminal, the client terminal is connected with the second server.

The accepting server of the present invention may be a server comprising: connection-order set means which, at the time of receiving a first connection request from the client terminals, sets the connection priority of the client terminal; and connection management means for allowing connection of the client terminals according to highest connection priority, at the time of receiving a second connection request from the client terminals after the first connection request.

The accepting server or other servers of the present invention may be further equipped with connection-number monitor means for monitoring a number of connectable client terminals. In this case, connection can be allowed to the client terminal with the highest connection priority, after acceptance of connection of a new client terminal has become possible, that is, when there is a vacancy in the first or second server.

The accepting server of the present invention can also transmit a program which automatically reruns or executes a connection request again, when refusing a connection request transmitted from the client terminal. Furthermore, when the accepting server of the present invention receives a connection request executed again based on the program by the client terminal, the accepting server may allow connection of the client terminal, based on the

connection priority of the client terminal. The aforementioned program can contain both data of the time interval, for causing the client terminal to execute a connection request again, and data for specifying a target connection server, that is, an accepting server.

In addition, the time interval for causing the client terminals to again execute a connection request can be set to a plurality of stages, based on the connection priorities of the client terminals, by an accepting server. For instance, the time interval for performing a connection request by the client terminal having a lower connection priority can be made longer than the time interval for a client terminal having a higher connection priority. Therefore, the number of connection requests can be suppressed as a whole. The time interval can be set to any number of stages.

In another accepting server of the present invention, the number of connectable client terminals is detected. The connection of the client terminal that made a connection request is allowed when the number of connectable client terminals is a reference value or greater. The connection of the client terminal is refused when the number of connectable client terminals is less than the reference value. Then, the connection priority of the client terminal is set when connection of the client terminal is refused. When the number of connectable client terminals is increased to more than the reference value, the right of connection is granted to the client terminal whose connection priority is highest. Next, the connection of the client terminal granted a right of connection is allowed when a connection request from the client terminal is received.

In the accepting server of the present invention, the connection queue data holding means generates a connection queue, based on the connection priority, and holds data of the connection queue.

Also, the connection-right data holding means holds data for identifying the client terminal granted a right of connection. The connection queue data holding means and the connection-right data holding means may be provided so that they are separated from each other. However, the connection queue data holding means and the connection-right data holding means may be provided in the same matrix, by providing an index or status information which discriminates between the client terminals in the connection queue and the client terminals granted a right of connection.

The connection allowance means of the present invention may make reference to the data held in the connection-right data holding means if it receives a connection request from the client terminal, and may allow connection of the client terminal if it is confirmed that a right of connection has been granted to the client terminal.

Note that information on the connection priority of the client terminal may be contained in "Cookie" data that is transmitted from the accepting server to the client terminal. With use of the "Cookie" data, the information on the connection priority is held even when the browser window of the client terminal is closed. In addition, connection priority may also be contained in data other than the "Cookie" data.

The present invention includes a connection-acceptance managing method comprising the steps of: receiving a request for connection to a server from an external client terminal; deciding, in response to the connection request, whether or not a connection to the server is allowed; setting a connection priority for the client terminal when the connection is refused; granting a right of connection to the client terminal which has the highest connection priority, each time a connection to the server becomes possible; and confirming whether or not a right of connection has been granted to the client terminal, when a connection request is

made by the client terminal for which the connection priority has been set, and allowing a connection of the client terminal to the server when a right of connection has been granted.

The present invention also includes a computer program that is executed by a computer which accepts requests of connection from client terminals, and a storage medium in which the computer program is stored so that the program is readable by the computer.

The program may further include a process of transmitting both data concerning the connection priority set for a client terminal, and a program for causing the client terminal to automatically execute a connection request again, after a predetermined time period, to the client terminal.

Furthermore, the present invention includes a computer program which causes a computer to execute a process of transmitting a connection request execution program, for automatically executing a connection request again, to the client terminal when a connection request from the client terminal is refused. The present invention also includes a storage medium in which the computer program is stored so that the program is readable by the computer.

Brief Description of the Drawings

Figure 1 is a block diagram showing a construction of a connection accepting system in a preferred embodiment of the present invention;

Figure 2 is a diagram showing a flow for a process of deciding whether a connection is established or not, when a connection request is received;

Figure 3 is a diagram showing a flow of a process for issuing a reference number to a client terminal;

Figure 4 is a diagram showing an example of data that is transmitted to the client terminal, the data including a program for automatically transmitting the connection request again to the server after a predetermined time period;

Figure 5 is a diagram showing an example of a screen, including information on a reference number which is displayed on the client terminal;

Figure 6 is a diagram showing another example of the screen that is displayed on the client terminal;

Figure 7 is a diagram showing a flow of a process for giving the client terminal a right of connection;

Figure 8 is a diagram showing how processing is performed when a connection request is received from the client terminal to which a reference number has been issued;

Figure 9 is a diagram showing how the server manages a connection queue;

Figures 10A and 10B are diagrams showing how a reference number is transmitted to the client terminal, the difference being shown between the case of hidden data (Figure 10A) and the case of "Cookie" data (Figure 10B);

Figure 11 is a diagram showing the case where a connection queue, and client terminals granted rights of connection, are arranged in the same matrix and discriminated with an index, when the server manages client terminals, based on reference numbers; and

Figure 12 is a diagram showing the case where a connection queue, and client terminals granted rights of connection, are arranged in the same matrix and discriminated with status information.

Detailed Description

The present invention will be described in detail with reference to a preferred embodiment shown in the accompanying drawings.

Figure 1 is a diagram for explaining the construction of a connection accepting system in the preferred embodiment of the present invention. In the figure, reference numeral 10 denotes a network such as the Internet, a local area network, and so forth, reference numeral 20 denotes a plurality of client terminals that users use, and reference numeral 30 denotes an accepting server (server) that accepts connection requests from the plurality of client terminals 20. Also, reference numeral 40 denotes an application server (another server) that executes a predetermined process in response to a request from the client terminal 20 where acceptance of the connection request has been completed in the accepting server 30.

Each client terminal 20 may include a personal computer (PC). The client terminal 20 has the function (or means) of receiving data transmitted from the accepting server 30 or application server 40, the function (or means) of executing a connection to the accepting server 30 or application server 40, and the browser function of displaying data transmitted from the accepting server 30 or application server 40.

The accepting server 30 is equipped with a connection management section 31 for executing a process of managing a connection in response to a connection request from the client terminal 20; a maximum-connection-number setting counter 32 to which the maximum

number of connections that can be connected simultaneously to the application server 40 is set; a connection-number counter 33 for indicating the number of client terminals 20 being connected to the application server 40, and a connectable-number counter (connection-number monitoring means or connectable-number detection means) 34 for indicating the number of client terminals 20 that can be connected to the applications server 40. In this arrangement, there is a relationship of "value in connectable-number counter 34" = "value in maximum-connection-number setting counter 32" - "value in connection-number counter 33" - "number of client terminals 20 whose reference number is present in connection-right acquired pool section 36 (to be described later)".

The connection management section 31 functions as connection management means and judgement means, and manages whether or not a connection is allowed in response to a connection request from the client terminal 20, based on a counter value in the connectable-number counter 34. The connection management section 31 allows the connection of the client terminal 20 with the application server 40 if a counter value in the connectable-number counter 34 is greater than or equal to a reference value of 1, and refuses the connection if the counter value is 0 less than the reference value.

The connection management section 31 also functions as priority-order set means and connection-order set means. That is, when it receives a first connection request, the connection management section 31 gives a reference number, for deciding the connection priority, to the client terminal 20 which has been refused a connection, and then manages the order of connection based on the reference number. To perform this management, the accepting server 30 is provided with a connection queue data holding section (or connection queue data holding means) 35 and the aforementioned connection-right acquired pool section (or

connection-right data holding means) 36. In this embodiment of the present invention, the connection queue data holding section 35 and the connection-right acquired pool section 36 are part of the memory region of Java Virtual Machine by which a program group written in Java language for controlling the accepting server 30 is operated.

The connection management section 31 generates a connection queue B in which data are arranged in the order of reference numbers, based on the reference numbers given to the client terminals 20, and causes the connection queue data holding section 35 to hold the generated data. Also, at the time that the application server 40 reaches a state in which a connection request from the client terminal 20 can be accepted (i.e., the counter value in the connectable-number counter 34 is 1 or greater), the connection management section 31 functions as the connection-right granting means in order to transfer the data of the reference number whose connection priority is the highest in the connection queue B, from the connection queue data holding section 35 to the connection-right acquired pool section 36. With this, the client terminal 20 whose data of the reference number is present in the connection-right acquired pool section 36 can be discriminated as the client terminal 20 allowed a connection right. Then, when there is a second connection request from the client terminal 20, if a reference number corresponding to that client terminal 20 is present in the connection-right acquired pool section 36, the connection management section 31 functions as connection allowance means in order to allow the connection of the client terminal 20 with the application server 40.

Furthermore, the connection management section 31 functions as program transmission means to transmit a connection-request execution program to the client terminal 20 given a reference number. The connection-request execution program is used for causing the client terminal 20 to again execute a connection

request to the accepting server 30, after a predetermined time period.

Now, the acceptance process in the accepting server 30 constructed as described above will be described in detail with reference to Figures 2 through 9.

When receiving a connection request from the client terminal 20 through the network 10, the accepting server 30 makes reference to the connectable-number counter 34 and decides whether or not the counter value is 1 or greater (step S101: decision process). As a result, in the case where the counter value is 1 or greater, the accepting server 30 accepts a connection request from the client terminal 20 (step S102) and adds 1 to the counter value (i.e., the number of connections) of the connection-number counter 33 (step S103).

Then, the connection management section 31 allows the client terminal 20 to connect with the application server 40 and transmits data, containing uniform resource locators (URL: data for specifying a target connection server), for connecting with the application server 40, to the client terminal 20. The client terminal 20 receiving this logs in to the connection management section 41 of the application server 40 shown in Figure 1 and requests an application processing section 42, which executes processing based on an application program, to perform a predetermined process (step S104).

In the case where, in step S101, the counter value of the connectable-number counter 34 is not 1 or greater, i.e., in the case where the counter value is 0, the acceptance process shifts to a reference-number issue process (step S200: priority order set process).

In the reference-number issue process, as shown in Figure 3, the

connection management section 31 of the accepting server 30 issues a reference number which indicates the connection priority to the client terminal 20 (step S201). The reference number to be issued will be satisfied if the accepting server 30 can identify the connection priority for the client terminal 20. In this embodiment, the reference number consists, for example, of "YYYYMMDD#n," where Y indicates year, M indicates month, D indicates date, and n indicates a number, issued in sequence on the day of its issue.

The connection management section 31 adds the issued reference number to the connection queue B held in the connection queue data holding section 35 (step S202).

In step S203, the accepting server 30 sends back the data of the reference number issued by the connection management section 31 to the client terminal 20. In sending back the reference number data, the accepting server 30 transmits the data of a character string representing the reference number, to the client terminal 20 as a "Cookie." The accepting server 30 also sends back data, responding to the connection request from the client terminal 20, for display on the browser screen of the client terminal 20.

As the data for displaying the response on the browser screen of the client terminal 20, this embodiment sends back, for example, data described in hypertext markup language (HTML), such as that shown in Figure 4. Based on this data, the browser screen of the client terminal 20 displays messages (information on the connection priority), such as "Crowded now," "You can log in in the order of ($Order) of ($Queue Size) people," "Your reference number is ($DTX_Ticket)," etc. The "$Queue Size" is the size of the connection queue B (the number of client terminals 20 which are waiting for connection) which is present in the connection queue data holding section 35. The "$Order" is the connection priority within the connection queue B. The "$DTX_Ticket" is a

reference number given by the "Cookie." On the actual browser screen, a numeral string, etc., output from the accepting server 30, is displayed. Figure 5 shows an example of a browser screen that may be used for displaying such messages.

Based on a source code, shown in the part A of Figure 4, of the data sent back from the accepting server 30, the client terminal 20 is loaded with this data and caused to process a reconnection request ("requestConnect") with respect to the accepting server 30 after a predetermined time period ("$Interval"). The word "$Interval" is a numerical value that is set by the accepting server 30, as described later. The "requestConnect()" process is the process of sending a connection request to "/servlet/ConnectServelt" (i.e., an example of the URL for the accepting server 30) which is data for specifying a target connection server, as shown in a source code in part B of Figure 4.

In this manner, in the aforementioned step 230, the accepting server 30 transmits to the client server 20, the program for automatically sending a connection request to the accepting server 30 after the time period ($Interval) set by the accepting server 30 has elapsed (transmission program, program transmission program). With this, the client terminal 20, given a reference number, automatically transmits a connection request again to the accepting server 30 after a predetermined time period.

In transmitting the data of the automatic connection request to the client terminal 20, as described above, the accepting server 30 sets a set time period ("$interval") for executing the automatic connection request. The value of this set period may be based on the connection priority, in the connection queue B of the connection queue data holding section 35, of the reference number issued to the client terminal 20. More specifically, the set time period ("$interval") may be, for example, 10 sec for reference

numbers whose connection priority is first to tenth in the connection queue B of the connection queue data holding section 35. Also, the set time period ("$interval") may be, for example, 30 sec for reference numbers whose connection priority is eleventh to fiftieth, and may be for example, 60 sec for reference numbers whose connection priority is fifty-first to one-hundredth.

In the case where the connection queue B in the connection queue data holding section 35 has reached its maximum queue size previously set, the accepting server 30 may transmit data for displaying a display screen such as that shown in Figure 6 to the client terminal 20, without issuing a reference number. In this case, the program for automatically transmitting a connection request again is not transmitted to the client terminal 20, so a connection request at the client terminal 20 will be manually repeated.

Note that the granularity of the set time period stages, the length of the set time period at each stage, and the priority order which becomes a threshold value for each stage, are not limited to the aforementioned numerical values. Also, depending on its performance, the accepting server 30 may transmit the reference number and the automatic transmission program for a connection request to all the client terminals 20, instead of displaying the display screen of Figure 6 without issuing the reference number.

As shown in Figure 7, in the case where the client terminal 20 ends its connection to the application server 40, the application server 40 informs the accepting server 30 that a connection right to the application server 40 has been released. If it is informed of the release of the connection right, the accepting server 30 first makes reference to the connection queue data holding section 35 and then confirms whether or not there is a connection queue B (step S301).

As a result, if there is no connection queue B, the counter value (i.e., the number of present connections) in the connection-number counter 33 is decremented by 1 and the counter value of the connectable-number counter 34 is incremented by 1 (step S302).

On the other hand, in the case where there is a connection queue B, the counter value (i.e., the number of present connections) in the connection-number counter 33 is decremented by 1. Also, the reference number in the connection queue B which has the highest connection priority is transmitted to the connection-right acquired pool section 36. With this, a connection right is granted to the client terminal 20 whose reference number has been transmitted to the connection-right acquired pool section 36 (step S303: connection-right grant process). Note that the reference number transmitted to the connection-right acquired pool section 36 is erased from the connection queue B, so each reference number in the connection queue B becomes higher in priority order of connection by 1.

If the accepting server 30 performs the aforementioned process, the client terminal 20 automatically executes a rerun of connection request with the accepting server 30 after a predetermined time period, based on the data transmitted in step S203 of Figure 3 (see Figure 4). At the time of this rerun of connection request, the data of the "Cookie" representing the reference number transmitted from the accepting server 30 to the client terminal 20 is attached and transmitted to the accepting server 30.

If the accepting sever 30 receives the rerun of the connection request from the client terminal 20, the connection management section 31 acquires the reference number contained in the character string in the data of the "Cookie" attached to the rerun of the connection request and confirms whether or not the

reference number is present in the connection-right acquired pool section 36, as shown in Figure 8 (step S401: connection-right confirmation process).

As a result, in the case where a reference number is present in the connection-right acquired pool section 36, the client terminal 20 is specified from the Cookie data and the connection of this client terminal 20 with the application server 40 is allowed. Then, the data that contains the URL for connecting with the application server 40 is transmitted to the client terminal 20. The client terminal 20 logs in to the connection management section 41 of the application server 40 (step S402: connection allowance process).

Next, the above-mentioned reference number is erased from the connection-right acquired pool section 36 (step S403), and the counter value (i.e., the number of present connections) of the connection-number counter 33 is incremented by 1 (step S404).

In the case where, in step S401, no reference number is present in the connection-right acquired pool section 36, the connection management section 31 transmits the automatic-retransmission data for a request to the accepting server 30 to the client terminal 20, as with the case shown in Figure 4 (step S405). Note that the "$QueueSize," "$Order," "$DTX_Ticket," and "$Interval" in the data to be transmitted are set to numerical values according to the status of the connection queue B of the connection queue data holding section 35 at that time.

With this, a connection request to the accepting server 30 can be automatically retransmitted from the client terminal 20 after a predetermined time period ($Interval), based on the automatic-retransmission data transmitted in step S405.

On the side of the client terminal 20, a cancel button A, may be

displayed on the browser screen, as shown in Figure 5, based on the data (see Figure 4) transmitted from the accepting server 30 in steps S203 of Figure 3 and S405 of Figure 8. When this cancel button A is operated at the client terminal 20, the process of "cancelRequest" is executed as shown in the part C of Figure 4. The "cancelRequest" process is the process of requesting the accepting sever 30 (/servlet/ConnectServlet: an example of URL) to cancel a reference number, as shown in the source code of the part D in Figure 4. The cancel request, along with the "Cookie" received from the accepting server 30, is transmitted to the accepting server 30.

In the accepting server 30 receiving the cancel request, the reference number obtained from the "Cookie" is removed from the connection queue data holding section 35.

Also, the accepting server 30 manages data held in the connection queue data holding section 35, as shown in Figure 9. Each time a previously set time period has elapsed (step S501), the accepting server 30 confirms whether or not a connection request has been transmitted again from the client terminal 20 corresponding to a reference number in the connection queue B held in the connection queue data holding section 35 (step S502).

As a result, if a connection request is rerun, the accepting server 30 returns to step S501 and repeats the same process. On the other hand, in the case where the connection request is not rerun within the set time period (e.g., in the case where the browser on the side of the client terminal 20 has been closed), the corresponding reference number is removed from the connection queue data holding section 35 (step S503).

In the aforementioned manner, the accepting server 30 receives a request to connect with the application server 40, made via the network 10 by the client terminal 20. The accepting server 30

also issues a reference number to the client terminal 20 when a first connection request is made. Based on this reference number, the connection priority with the application server 40 is managed. Furthermore, when issuing the reference number, the accepting server 30 transmits the program for automatically sending a connection request after a predetermined time period, to the client terminal 20. Then, when the client terminal 20 executes the second connection request, the client terminal 20 can automatically connect with the application server 40 if it has acquired a connection right. The client terminal 20 can repeat the connection request again after a predetermined time period, if it has not acquired a connection right.

This eliminates the necessity for the client terminal 20 to repeat a connection request by trial and error. Thus, the processing load on the side of the accepting server 30 can be alleviated. In addition, it becomes possible to perform the process reliably and quickly and the management of connections can be performed strictly according to priorities that correspond to order of arrival.

In addition, the accepting server 30 sets a time period for the next connection request in accordance with the connection priority in the connection queue B and also makes a time period for the client terminal 20 having a lower connection priority longer than that for the client terminal 20 having a higher connection priority. Therefore, the number of connection requests can be reduced as a whole and the processing load on the accepting server 30 can be reduced.

Furthermore, the accepting server 30 and the application server 40 are independently provided so that only the client terminal 20 that is allowed to be connected by the accepting server 30 is connected to the application server 40. Thus, the application server 40 does not need to process connection requests and is

capable of performing the original process more efficiently.

Furthermore, the user can see his status while waiting for a connection with the application server 40, because information such as a connection priority in the connection queue B, a reference number, etc., is displayed on the browser screen of the client terminal 20, based on the data transmitted from the accepting server 30.

At this time, if the accepting server 30 predicts the time needed for acquiring a right of connection, based on the average processing time in the application server 40, it is also possible to display this on the browser screen of the client terminal 20.

While, in the above-mentioned embodiment, information such as a reference number is displayed on the browser screen of the client terminal 20, the contents of the information can be changed as appropriate. It is not necessary, however, not to display this information.

The reference number may be conveyed by any means, so long as the accepting sever 30 can specify the connection priority given to the client terminal 20. For example, the reference number may contain time in addition to year, month, and day. Also, a character string, or other symbolic identifier may be employed instead of a numerical string.

The accepting server 30 may issue a reference number in the form of "Cookie" to the client terminal 20. However, the reference number may instead be transmitted as data transmittable to the accepting server 30, such as hidden data, contained in the HTML data which is transmitted from the accepting sever 30 to the client terminal 20, URL embedding a reference number, etc. Also, it can be transmitted as a part of a program for generating transmission data by use of script language such as JavaScript,

VBScript, etc. Furthermore, it can be transmitted as data that is used for making reference to a program for generating transmission data. In this case, as shown in Figure 10A, the same client terminal 20 can receive issue of a plurality of reference numbers, if it opens a plurality of browser windows and executes a connection request to the accepting server 30 through the respective browser screens. In the case of "Cookie" data, as shown in Figure 10B, only a single reference number can be acquired, even if a plurality of browser windows are employed. In the case of employing hidden data, however, data for a reference number will be lost, if the browser window is closed or overwritten with another document. On the other hand, in the case employing "Cookie" data shown in the above-mentioned embodiment, there is no possibility that a reference number will be lost, as long as the term of validity for the "Cookie" does not expire.

While the data for automatically resending a connection request may be transmitted in the form of HTML from the accepting server 30 to the client terminal 20, it also may be transmitted in other languages or data forms, such as extensible markup language(XML). More specifically, the above-mentioned embodiment is also applicable, for example, when using a program operable within a browser such as an applet program as the client terminal 20, and the case of using a program, other than a browser program, which communicates with the accepting server 30 as the client terminal 20. In that case, the data that is transmitted from the accepting server 30 to the client terminal 20 can also be transmitted as data inherent to an application program rather than data described, for example, in HTML language or XML language.

In the above-mentioned embodiment, the connection queue data holding section 35 holds the data of the connection queue B, and the connection-right acquired pool section 36 holds the data of a reference number having a right of connection. However, the data held by the holding section 35 and the data held by the pool

section 36 can be combined and managed in a single matrix, as shown in Figure 11. That is, a matrix C may be generated so that reference numbers are arranged in the order of connection priority and that an index I indicating the leading end of the connection queue B is provided between the connection-right acquired group and the connection queue B. In this case it becomes possible to perform management with only a single matrix.

As shown in Figure 12, as with Figure 11, a matrix D may be generated so that reference numbers are arranged in the order of connection priority and given status information indicating whether or not there is a right of connection.

In the above-mentioned embodiment, the connection queue data holding section 35 and the connection-right acquired pool section 36 are held in the memory region of Java Virtual Machine on which Java programs for controlling the accepting server 30 work. However, the data held by these sections can also be stored in a database. Similarly, the data for the matrixes C and D shown in Figures 11 and 12 can be stored in a database. However, in the case of utilizing the database, there are cases where throughput will be reduced compared with the case of holding data in the memory region of Java, because it is necessary to read out or write data from or to the database. Note that in the case of holding data in a memory region that is utilized by a program, language other than Java may be utilized as programming language.

The accepting server 30 may be constructed in various ways, as long as the functions shown in the above-mentioned embodiment are realized. For instance, the accepting server 30 may adopt the common gateway interface (CGI) most widely used, or can also adopt a servlet that can be expected to enhance throughput.

The accepting server 30 and the application server 40 can also be realized as a single server, if they are functionally separated

from each other. In order to enhance the entire throughput, the accepting server 30 may be provided with a plurality of application servers 40. Also, there may be a plurality of pairs of an accepting server 30 and an application server 40. However, in this case it is necessary that each pair control reference numbers independently and that the client terminal 20 also send a connection request to the accepting server 30 of the same pair. On the other hand, if the function of generating a reference number, the connection queue data holding section 35, and the connection-right acquired pool section 36 are separated from the accepting server 30 to construct a new server, and this new server is connected with a plurality of accepting servers 30 and application servers 40, the new server can also manage reference numbers collectively while enhancing the entire system throughput.

The program for issuing a reference number at the time of receiving a connection request and for managing a connection based on the reference number, the program for transmitting the data for the automatic retransmission of the connection request to the client terminal 20, the program for setting different time intervals for the automatic retransmission, and so forth, can also be achieved by the following storage media and program transmission units.

That is, the above-mentioned programs that are executed by a computer can be stored in a storage medium, such as a CD-ROM, a DVD, memory, a hard disk, etc., so that the computer can read out the programs.

The program transmission unit may be provided with storage means, such as a a CD-ROM, a DVD, memory, a hard disk, etc., in which the aforementioned programs are stored, and transmission means for reading out the programs from the storage means, and transmitting the programs to a unit which executes the programs, through connectors, or a network such as the Internet, a local area

network, and so forth. Such a program transmission unit is particularly suitable for installing the aforementioned programs in the accepting server 30.

As has been described above, the present invention is capable of reducing a processing load on the server that receives a request for connection from a client terminal, executing processing quickly and reliably, and acceptance connection requests in the order of arrival.

While the present invention has been described with reference to the preferred embodiment thereof, the invention is not limited to the details given herein, but may be modified within the scope of the invention claimed.